NIST Special Publication 800-107

DRAFT

Recommendation for Using Approved Hash Algorithms

Quynh Dang

Computer Security Division
Information Technology Laboratory

C O M P U T E R S E C U R I T Y

July 2007

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

Technology Administration
Robert Cresanti, Under Secretary of Commerce for Technology

National Institute of Standards and Technology
William Jeffrey, Director

NIST **National Institute of Standards and Technology** • Technology Administration • U.S. Department of Commerce

Abstract

Cryptographic hash functions that compute a fixed- length message digest from arbitrary length messages are widely used for many purposes in information security. This document provides security recommendations for using the hash functions Approved in Federal Information Processing Standard (FIPS) 180-3 for different applications.

KEY WORDS: digital signatures, hash algorithms, hash functions, hash-based key derivation algorithms, hash value, HMAC, message digest, randomized hashing, random number generation, SHA, truncated hash values.

Acknowledgements

The author, Quynh Dang of the National Institute of Standards and Technology (NIST) gratefully appreciates an enormous reviewing work by Elaine Barker. The author also acknowledges great inputs from John Kelsey, W. Timothy Polk, William E. Burr, Shu-jen Chang and Donna F. Dodson in development of this Recommendation.

Table of Contents

Recommendation for Using Approved Hash Algorithms

1 Introduction

A hash algorithm is used to condense messages of arbitrary length to a smaller, fixed-length message digest. Federal Information Processing Standard (FIPS) 180-3, the Secure Hash Standard (SHS) [FIPS 180-3], specifies five Approved hash algorithms: SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512. Secure hash algorithms are typically used with other cryptographic algorithms.

This Recommendation provides guidance on using the Approved hash algorithms in digital signatures [FIPS 186-3], HMACs [FIPS 198-1], key derivation functions (KDFs) and random number generators.

2 Authority

This Recommendation has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This recommendation is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This Recommendation has been prepared for use by Federal agencies. It may be used by non-governmental organizations on a voluntary basis and is not subject to copyright (attribution would be appreciated by NIST.)

Nothing in this Recommendation should be taken to contradict standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should this Recommendation be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official.

Conformance testing for implementations of this Recommendation will be conducted within the framework of the Cryptographic Module Validation Program (CMVP), a joint effort of NIST and the Communications Security Establishment of the Government of Canada.

3 Glossary of Terms, Acronyms and Mathematical Symbols

3.1 Terms and Definitions

Algorithm	A clearly specified mathematical process for computation; a set of rules that, if followed, will give a prescribed result.
Approved	FIPS-Approved and/or NIST-recommended. An algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, or 2) adopted in a FIPS or NIST Recommendation or 3) specified in a list of NIST Approved security functions.
Approved hash algorithms	Hash algorithms specified in [FIPS 180-3].
Bit string	An ordered sequence of 0 and 1 bits. The leftmost bit is the most significant bit of the string. The rightmost bit is the least significant bit of the string.
Bits of security	See security strength.
Block cipher	A symmetric key cryptographic algorithm that transforms a block of information at a time using a cryptographic key. For a block cipher algorithm, the length of the input block is the same as the length of the output block.
Collision	Two different known messages that have the same message digest.
Digital signature	The result of applying some cryptographic functions to data that, when the functions are properly implemented, provides origin authentication, data integrity and signatory non-repudiation.
Entropy	A measure of the disorder, randomness or variability in a closed system. The entropy of X is a mathematical measure of the amount of information provided by an observation of X.
Hash algorithm	See hash function.
Hash function	A function that maps a bit string of arbitrary length to a fixed length bit string. Approved hash functions are specified in [FIPS 180-3].
Hash output	See “message digest”.
Hash value	See “message digest”.

Key	A parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce the operation, while an entity without knowledge of the key cannot. Examples applicable to this Recommendation include: 1. The computation of a keyed-hash message authentication code, and 2. The verification of a keyed-hash message authentication code. 3. The generation of a digital signature of a message. 4. The verification of a digital signature.
Keying material	The data (e.g., keys and IVs) necessary to establish and maintain cryptographic keying relationships.
Message authentication code	A bit string of fixed length, computed by a MAC generation algorithm, that is used to establish the authenticity and, hence, the integrity of a message.
Message digest	The result of applying a hash function to a message. Also known as a "hash value" or "hash output".
Random bit generator	A device or algorithm that can produce a sequence of random bits that appears to be statistically independent and unbiased.
Randomized hashing	A process by which the input to a hash function is randomized before being processed by the hash function.
Random number	A value in a set that has an equal probability of being selected from the total population of possibilities and, hence, is unpredictable. A random number is an instance of an unbiased random variable, that is, the output produced by a uniformly distributed random process.
Security strength	A number associated with the amount of work (that is, the number of operations or work factor (see work factor)) that is required to break a property of a cryptographic algorithm or system; a security strength is specified in bits.
Shall	Used to indicate a requirement of this Recommendation.
Shared secret	A secret value that has been computed using a key agreement scheme and is used as input to a key derivation function.
Work factor	A number of executions of an algorithm by an adversary that is required to break some property of the algorithm. For example, for SHA-256, a work factor of 2^{128} executions of the algorithm is required by an adversary to find a collision.

3.2 Acronyms

FIPS	Federal Information Processing Standard
SHA	Secure Hash algorithm
KDF	Key Derivation Function
HKDF	Hash-based Key Derivation Function
MAC	Message Authentication Code
HMAC	Keyed-hash Message Authentication Code

3.3 Symbols

K	HMAC key.
L	The length of the hash value in bytes.
$L/2$	L is divided by 2.
λ	The length in bits of a truncated MAC or truncated hash value.
T	The length in bytes of a truncated MAC.

4 Approved Hash Algorithms

Currently, there are five Approved hash algorithms, SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512, which are specified in [FIPS 180-3]. These hash algorithms produce outputs of 160, 224, 256, 384 and 512 bits, respectively. The output of a hash algorithm is commonly known as a message digest, a hash value or a hash output.

A hash function[1] is used to produce a message digest that has one or more of the following three properties:

1. Collision resistance: It is computationally infeasible to find two different inputs to the hash function that have the same hash value. That is, if *hash* is a hash function, it is computationally infeasible to find two different inputs *x* and *x'* for which *hash*(*x*) = *hash (x')*. Collision resistance is measured by the number of hashing operations (i.e., the amount of work; the work factor) that would be needed to find a collision for a hash function. The amount of collision resistance provided by a hash-function cannot exceed half the length of the hash value produced by a given hash function. For example, SHA-256 produces a (full-length) hash value of 256 bits; SHA-256 cannot provide more than 128 bits of collision resistance.

[1] Hash function and hash algorithm are used interchangeably, depending on the context of the discussions throughout this Recommendation.

Note: the strength of some property of a cryptographic algorithm is determined from the work factor required to break the property; if the work factor is 2^x, then the strength is defined to be *x* bits.

2. Preimage resistance[2]: Given a randomly chosen hash value, *hash_value*, it is computationally infeasible to find an *x* so that *hash(x) = hash_value*. This property is also called the one-wayness property. Preimage resistance is measured by the amount of work that would be needed to find a preimage for a hash function. The amount of preimage resistance provided by a hash-function cannot exceed the length of the hash value produced by a given hash function. For example, SHA-256 cannot provide more than 256-bits of preimage resistance; this means that a work factor of 2^{256} operations will likely find a preimage of a (full-length) SHA-256 hash value.
3. Second preimage resistance: It is computationally infeasible to find a second input that has the same hash value as any other specified input. That is, given an input *x*, it is computationally infeasible to find a second input *x'* that is different from *x*, such that *hash*(*x*) = *hash* (*x'*). Second preimage resistance is measured by the amount of work that would be needed to find a second preimage for a hash function. The amount of second preimage resistance provided by a hash-function cannot exceed the length of the hash value produced by a given hash function. For example, SHA-256 cannot provide more than 256-bits of second preimage resistance.

The security strength of a hash function for digital signatures is defined as its collision resistance strength. Not all applications of hash functions require collision resistance, but may require preimage and/or second preimage resistance. A hash function that is not suitable for a digital signature application might be suitable for other cryptographic applications that do not require collision resistance. The security strengths of Approved hash functions for different applications can be found in [SP 800-57].

An image (i.e., a hash value) always has a corresponding preimage; a preimage is the input to the hash function that produces the given hash value. An image might or might not have a second preimage which is a different input to the hash function that produces the same hash value. If a second preimage exists, then there is no way to determine which preimage (the real preimage or the second preimage) actually produced the given hash value; either of the preimages could be the authentic one. If a hash function is preimage resistant, then it is also second preimage resistant.

4.1 Strengths of the Approved Hash Algorithms

If strength (security strength) of some property of a cryptographic algorithm is n bits, a work factor of 2^n is required to break the property of the cryptographic algorithm with 100% probability of success. And, a work factor of 2^m ($1 <= m < n$) will break the

[2] There are slightly different definitions of preimage resistance of hash functions in the literature.

property with a success probability of $2^{(m-n)}$. For instance, collision resistance of the cryptographic algorithm SHA-224 (see Table 1 below in this section) has the strength of 112 bits (n = 112); a work factor of 2^{112} is required to break the property of the algorithm with 100% probability of success. And, a work factor of 2^{100} (m = 100) will break the collision resistance property of the algorithm with $2^{(100-128)}$ ($2^{(-28)}$) probability of success.

As mentioned above, the collision resistance strength of any Approved hash function cannot exceed half the length of its hash value. Currently, SHA-224, SHA-256, SHA-384 and SHA-512 are believed to have collision resistance strengths of 112, 128, 192 and 256 bits (half of the lengths of their hash values), respectively. However, due to the latest cryptanalytic results for SHA-1, SHA-1 may have a collision resistance strength that is considerably less then half of its hash value.

The preimage resistance strengths of SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 are 160, 224, 256, 384 and 512 bits (the lengths of the hash values) (see Table 1), respectively. At the time that this Recommendation was written, there are no known short cuts to find the preimages of the hash values generated from the Approved hash algorithms.

Second preimage resistance strengths of the Approved hash functions depend not only on the functions themselves, but on the sizes of the messages that the hash functions process as well. The second preimage resistance strength of any of the Approved hash algorithms is ($L - K$), where L is the output block size of the hash algorithm, for instance $L = 256$ in SHA-256, (i.e., the length of the hash value), and K is a function of the input block size, for instance input block size = 512 for SHA-256 (See more in [FIPS 180-3]) and the length of the message used (the preimage) as follow:

$$2^{K} = \frac{message\_length\_in\_bits}{input\_block\_size\_in\_bits}$$

Note that the second preimage resistance strengths of the Approved hash functions are not fixed values; they depend on the environments where the hash functions are used.

Table 1 provides a summary of the strengths of the Approved hash algorithms:

Table 1: Summary of the Strengths of the Approved Hash Algorithms

	SHA-1	**SHA-224**	**SHA-256**	**SHA-384**	**SHA-512**
Collision Resistance Strength in bits	< 80	112	128	192	256
Preimage Resistance Strength in bits	160	224	256	384	512
Second Preimage Resistance Strength in bits	160 - *K*	224 - *K*	256 - *K*	384 - *K*	512 - *K*

Also, note that the preimage resistance of any Approved hash algorithm is stronger than its collision resistance strength. Therefore, if an Approved hash algorithm satisfies a collision resistance requirement for a cryptographic application, then it also satisfies any preimage resistance requirement that the application might have.

Similarly, if *K* is not greater than *L/2* (half of the output block size in bits) or $(L - K) \geq L/2$, then the second preimage resistance strength of the Approved hash algorithm (or function) is equal to or greater than its collision resistance strength. In this case, if the Approved hash function satisfies a collision resistance requirement for a cryptographic application, then it also satisfies any second preimage resistance requirement that the application might have. A value of *K* that is less than or equal to *L/2* is very common, in practice. For example, if an Approved hash function in a digital signature application satisfies the collision resistance requirement, and the messages hashed by the application is not longer than $2^{(L/2)}$ input blocks of the hash function in length, then it also satisfies the second preimage resistance requirement.

5 Hash Function Usage

5.1 Truncated Hash Value

Some applications may require a message digest that is shorter than the (full-length) message digest provided by an Approved hash function specified in [FIPS 180-3]. In such cases, it may be appropriate to use a subset of the bits produced by the hash function as the (shortened) message digest.

This guidance is provided for unforeseen applications in order to provide interoperability. A standard method for truncating hash function outputs (i.e., message digests) is provided strictly as a convenience for implementers and application developers. The proper use of a truncated hash value is an application-level issue.

For convenience, let the shortened message digest be called a truncated hash value, and let λ be the desired length in bits of the truncated hash value. Let *hlen* be the length in bits of the (full-length) message digest for a given hash function. A truncated hash value may be used if the following requirements are met:

1. λ shall be at least twice the required collision resistance strength s (in bits) for the truncated hash value (i.e., $\lambda \geq 2s$).
2. The length of the output block of the Approved hash function to be used shall be greater than λ (i.e., $hlen > \lambda$).
3. The λ left-most bits of the full-length message digest shall be selected as the truncated hash value.

For example, if a truncated hash value of 96 bits is desired, the SHA-256 hash function could be used (e.g., because it is available to the application, and provides an output larger than 96 bits). The leftmost 96 bits of the 256-bit message digest generated by the SHA-256 hash function are selected as the truncated hash value, and the rightmost 160 bits of the message digest are discarded. Truncating the message digest can impact the security of an application. The amount of collision resistance does not exceed half the length of the truncated hash value. Even though SHA-256 provides 128 bits of collision resistance, the collision resistance provided by the 96-bit truncated hash value is limited to half the length of the truncated hash value, which is 48 bits, in this case.

Truncating the message digest can have other impacts, as well. For example, applications that use a truncated hash value risk attacks based on confusion between different parties about the specific amount of truncation used, and the specific hash function that was used to produce the truncated hash value. Any application using a truncated hash value is responsible for ensuring that the truncation amount and the hash function used are known to all parties, with no chance of ambiguity. It is also important to note that there is no guarantee that truncation will not make any Approved hash functions weaker.

5.2 Digital Signatures

A hash function is used to condense a message to a fixed-length message digest. For digital signature generation, this message digest is then signed by a digital signature algorithm. A digital signatures is used to verify the data that was signed and who signed it.

When two different messages have the same hash value or message digest (i.e., a collision is found), then a digital signature of one message may be used as a digital signature for the other message. If this happens then a verified digital signature does not guarantee the authenticity of the signed data because either one of the two messages could be valid. Therefore, hash functions used in digital signature applications shall be collision resistant.

The collision resistance strengths of the hash functions are used as their security strengths for digital signature applications. The collision resistance strengths of the Approved hash algorithms can be found in Table 1 of Section 4.1 of this Recommendation. The security strength of any digital signature algorithm specified in [FIPS 186-3] is the minimum of the collision resistance strength of the hash algorithm and the security strength provided by the digital signature algorithm and key size; more information can be found in [SP 800-57, Part 1]). For instance, if a digital signature that is generated using SHA-1 and one of the Approved digital signature algorithms and key sizes specified in [FIPS 186-3], then the security strength of this digital signature is equivalent to the collision resistance strength of SHA-1, which is less then 80 bits (see Table 1 in Section 4.1). Note that the security strength of any Approved digital signature algorithm and key size is at least 80 bits. Therefore, SHA-1 should not be used in any digital signature applications that require at least 80 bits of security. More information on the security strengths of digital signature applications using the Approved hash algorithms can be found in [SP 800-57].

There are several ways to use hash functions with digital signature algorithms as described below.

5.2.1 Full-length Hash Values

The hash functions specified in [FIPS 180-3] (i.e., SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512) generate (full-length) hash values of 160, 224, 256, 384 and 512 bits, respectively. Full-length hash values as specified in [FIPS 180-3] can be used with Approved digital signature algorithms as specified in [FIPS 186-3]. The amount of collision resistance provided by a hash-function does not exceed half the length of the full-length hash value.

5.2.2 Truncated Hash Values

Truncated hash values may be used in generating digital signatures. However, the security of the hash functions now depends on the lengths of the truncated hash values, as well as the hash function that is used. As stated in Section 5.1, the amount of collision resistance does not exceed half the length of the truncated hash value.

The length of truncated hash values used shall be at least twice the desired amount of collision resistance (i.e., the security strength) required for the digital signature algorithms. For example, if security strength of 112 bits is required, a truncated hash value of at least 224 bits must be produced. SHA-224, SHA-256, SHA-384 and SHA-512 could be used to generate a 224-bit hash value, although, in the case of SHA-224, the hash-value would not be truncated.

5.2.3 Randomized Hashing for Digital Signatures

As described in Section 5.2, the security strengths of Approved hash algorithms in digital signature applications [FIPS 186-3] are their collision resistance strengths. However, using the randomized hashing technique as specified in [SP 800-106], the security strengths of the Approved hash algorithms in the digital signature applications will be their second preimage resistance strengths, not their collision resistance strengths. The second preimage resistance strengths of the Approved hash algorithms are normally stronger than their collision resistance strengths (see the discussion in Section 4.1).

As stated in Section 4.1, SHA-1 may have a collision resistance strength less than 80 bits. Therefore, SHA-1 may not be suitable for 80 bits of security digital signature applications. However, SHA-1 has a second preimage resistance strength more than 80 bits when $K < L/2$ (see the discussion in Section 4.1), so SHA-1 will be suitable for applications requiring 80 bits of security when the randomized hashing technique specified in SP 800-106 is used.

5.3 Keyed-Hash Message Authentication Codes (HMAC)

5.3.1 Description

Message authentication codes (MACs) provide data authentication and integrity protection. Two types of algorithms for computing a MAC have been Approved: 1) MAC algorithms that are based on Approved block cipher algorithms (more information can be found in [SP 800-38B] and 2) MAC algorithms that are based on hash functions, called HMAC algorithms that are specified in [FIPS 198-1]. This section discusses the use of HMAC.

HMAC requires the use of a secret key that is shared between the entity that generates the MAC (e.g., a message sender), and the entity (or entities) that need to verify the MAC (e.g., a message receiver).

The MAC is formed by condensing the secret key and the data to be MACed (e.g., a message to be sent) using a MAC algorithm (e.g., HMAC). The MAC is typically provided to the MAC verifier along with the data that was MACed (e.g., the sender transmits both the MAC and the data that was MACed to the intended receiver).

The verifier computes the MAC on the data using the same key and MAC algorithm that was used by the MAC generator and compares the result computed with the received MAC. If the two values match, the message has been correctly received and the verifier is assured that the entity that generated the MAC is a member of the community of users that share the key.

If an adversary, who is not a member of the community of users that share the key, finds preimages of hash values in operations while computing a certain HMAC value, then he/she will know the shared key, because one of the preimages contains it. If this happens, then a HMAC value will fail to provide user authentication for future HMAC values if the same shared key is used, because the HMAC values could be generated by

the adversary, who knows the shared key, but does not in the community of users. Therefore, HMAC applications shall use hash functions that provide sufficient preimage resistance for the desired security strength of the HMAC. And, the security strengths of hash functions in HMAC applications are their preimage resistance strengths (see Table 1 of Section 4.1).

5.3.2 The HMAC Key

The security strength of HMAC depends on the preimage resistance strength of the hash function as described above and the security strength of the secret key. The key, *K,* shall contain at least $L/2$ bytes (i.e., $8 * L/2$ bits) of entropy, so that at least $(8* L/2)$ bits of security are provided, where L is the length (in bytes) of the hash function output. Note that secret keys that contain greater than L bytes ($8*L$ bits) of entropy do not significantly increase the security strength, due to the fact that the preimage resistance strength of the hash function is limited to the length in bits of the hash value.

HMAC keys shall be chosen at random using an Approved key generation or key establishment method and shall be changed periodically. The keys shall be protected in a manner that is consistent with the value of the data that is to be protected (i.e., the *text* that is authenticated using HMAC). Approved key generation methods include the generation of random bits using an Approved random bit generator as specified in [SP 800-90], Recommendation for Random Number Generation Using Deterministic Random Bit Generators, and the use of an Approved key establishment method (e.g., a method specified in [SP 800-56A], Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography).

5.3.3 Truncation

Applications may truncate the output of HMAC to produce a truncated value of T bytes (i.e., or λ bits, where $\lambda = 8*T$).

When a truncated hash value is used, and the length of the MAC needed is T bytes, then the T left-most bytes (i.e., the λ left-most bits) of the HMAC output shall be used as the MAC. The output length, T, shall be no less than four bytes (i.e., $4 \leq T \leq L$). However, T shall be at least $L/2$ bytes (i.e., $L/2 \leq T \leq L$) unless an application or protocol makes numerous trials impractical. For example, a low bandwidth channel might prevent numerous trials on a 4 byte MAC, or a protocol might allow only a small number of invalid MAC attempts. See Section 5.3.4 for further discussion.

5.3.4 Security Limitation of HMACs

The successful verification of a MAC does not completely guarantee that the accompanying data is authentic; there is a chance that an adversary with no knowledge of the key can present a purported MAC on the plaintext data that will pass the verification procedure. For example, an arbitrary purported MAC of λ bits on an arbitrary plaintext message may be successfully verified with an expected probability of $(1/2)^{\lambda}$. This limitation is inherent in any MAC algorithm.

The limitation is magnified if an application permits a given non-authentic message to be repeatedly presented for verification with different purported MACs. Each individual trial succeeds only with a small probability, $(1/2)^{\lambda}$; however, for repeated trials, the probability increases that, eventually, one of the MACs will be successfully verified. Similarly, if an application permits a given purported MAC to be presented with different non-authentic messages, then the probability increases that, eventually, the MAC will be successfully verified for one of the messages.

Therefore, in general, if the MAC is truncated, then its length, λ, should be chosen as large as is practical, with at least half as many bits as the output block length, $8*L$ (in bits). The minimum value for λ may be relaxed to 32 bits for applications in which the two types of repeated trials that are described above are sufficiently restricted. For example, the application, or the protocol that controls the application, may monitor all of the plaintext messages and MACs that are presented for verification, and permanently reject any plaintext message or any MAC that is included in too many unsuccessful trials.

Another example occurs when the bandwidth of the communications channel is low enough to preclude too many trials of either type. In both cases, the maximum number of allowed unsuccessful trials must be pre-determined, based on the risks associated with the sensitivity of the data, the length of λ and the MAC algorithm used; in the case of HMAC, this includes the hash function that is used.

5.4 Hash-based Key Derivation Functions (HKDFs)

Hash functions can be used as building blocks in key derivation functions (KDFs) (e.g., as specified in [SP 800-56A] for key establishment). KDFs using hash functions as their building blocks are called Hash-based Key Derivation Functions (HKDFs). Main purpose of a HKDF is to generate secret keys from a secret value that is shared between communicating parties. A compromise of one of the generated secret keys shall not lead to compromise of the shared secret value. If a hash function in the HKDF is not preimage resistant, then an adversary that knows one of the generated secret keys may be able to determine the shared secret value by finding the preimage(s) of the hash value(s) generated by the hash function. Therefore, to be able to have a secure HKDF, the hash function shall be preimage resistant.

Each of the two Approved HKDFs in [SP 800-56A] uses a shared secret that is shared between two communicating parties, an Approved hash function and other input attributes to generate secret keying material, such as HMAC keys. The security of an HKDF specified in [SP800-56A] depends on the preimage resistance strength of the hash function and the security strength of the shared secret generated from the key agreement scheme. The security of the keying material generation process is the minimum of the preimage resistance strength of the hash function (see Table 1 of Section 4.1), and the security strength of the shared secret. For instance, if an HKDF built with SHA-224 uses a shared secret with 80 bits of entropy (i.e., the shared secret provides 80 bits of security),

this HKDF has a security strength of 80 bits, even though SHA-224 provides a preimage resistance strength of 224 bits.

HKDFs are used to generate secret keying material for cryptographic functions, such as HMAC. Therefore, the security strength of the generated keying material is vital to the security of the cryptographic functions that use it. The security strength of an HKDF shall be equal to or greater than the desired security strength of any application that uses the generated keying material. For instance, if 128 bits of security strength are desired for an application, then the keying material shall be generated from a shared secret and an HKDF that provides at least 128 bits of security, i.e., the shared secret key shall have at least 128 bits of entropy, and the preimage resistance strength of the hash function shall be at least 128 bits. The hash function shall be one of the Approved hash functions in this case.

5.5 Random Number Generation

When a random number is needed by an application, random bits are generated by a random bit generator and then converted to a random number. However, some applications may require random bits rather than random numbers; in this case, the random bits need not be converted. Random bit generators may be constructed using hash functions. Approved random bit generators and the requirements for their use are specified in [SP 800-90].

6 References

[SP 800-38B] NIST Special Publication (SP) 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005.

[SP 800-56A] NIST Special Publication (SP) 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, March 2006.

[SP 800-57] NIST Special Publication (SP) 800-57, Part 1, Recommendation for Key Management: General, August 2005.

[SP 800-90] NIST Special Publication (SP) 800-90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, June 06.

[SP 800-106] NIST Special Publication (SP) 800-106, Randomized Hashing for Digital Signatures, Draft July 2007.

[FIPS 180-3] Federal Information Processing Standard 180-3, Secure Hash Standard (SHS), Draft June 2007.

[FIPS 186-3] Federal Information Processing Standard 186-3, Digital Signature Standard (DSS), Draft March 2006.

[FIPS 198-1] Federal Information Processing Standard 198-1, The Keyed-Hash Message Authentication Code (HMAC), Draft June 2007.